IAMNN: ITERATIVE AND ADAPTIVE MOBILE NEURAL NETWORK FOR EFFICIENT IMAGE CLASSIFICATION

Sam Leroux*, **Pieter Simoens & Bart Dhoedt**
Ghent University - imec, IDLab
Department of Information Technology
Technologiepark-Zwijnaarde 15
B-9052 Ghent, Belgium

Pavlo Molchanov, Thomas Breuel & Jan Kautz
NVIDIA
2788 San Tomas Expressway
Santa Clara, California
USA

ABSTRACT

Deep residual networks (ResNets) made a recent breakthrough in deep learning. The core idea of ResNets is to have shortcut connections between layers that allow the network to be much deeper while still being easy to optimize avoiding vanishing gradients. These shortcut connections have interesting side-effects that make ResNets behave differently from other typical network architectures. In this work we use these properties to design a network based on a ResNet but with parameter sharing and with adaptive computation time. The resulting network is much smaller than the original network and can adapt the computational cost to the complexity of the input image.

1 INTRODUCTION AND RELATED WORK

After their impressive results on the ILSVRC2015 challenge, deep residual networks (He et al., 2016) quickly became one of the default architectures for computer vision tasks. Instead of just stacking layers on top of each other where each layer has to transform the output of the previous layer ($h_{i+1} = F_i(h_i)$), they add *skip connections*, identity mappings that copy the input of the layer to the output. The layer then learns a *residual* to add to the input ($h_{i+1} = h_i + F_i(h_i)$) . This makes it easier to optimize the network and allows us to build networks with hundreds or even thousands of layers. ResNets are closely related to Highway networks (Srivastava et al., 2015) where the flow of information is regulated by gates instead of using a fixed skip connection.

The residual connections have some very interesting properties in addition to increasing the maximum depth of the network. Various works have shown that ResNets are remarkably robust against deleting or reordering layers from a trained network (Veit et al., 2016) (Srivastava et al., 2015) while this behavior is not found in the more traditional architectures. Veit et al. (2016) argue that ResNets should be interpreted as an exponential ensemble of shallower models. Because of the shortcut connections there is a large number of possible paths with different lengths and although all these paths are trained simultaneously they do not strongly depend on each other. Deleting some layers of the network corrupts certain paths but since there is an exponentially large number of paths the overall effect on the final accuracy is limited. Greff et al. (2016) on the other hand argue that the layers in a ResNet do not learn completely new representations but instead they gradually refine the features extracted by the previous layers. In this *unrolled iterative estimation view* successive layers cooperate to compute a single level of representation. Jastrzebski et al. (2017) provided a formal view of iterative feature refinement and showed that a each residual layer refines the feature representation to reduce the loss with respect to the hidden representation.

In this work we follow the iterative estimation view and exploit these properties to design networks with less parameters and a smaller computational cost. To reduce the model size we enforce a strict parameter sharing between the layers in the network. Since each layer refines the features extracted by the previous layer instead of learning completely new features, it seems plausible that there is a certain redundancy in having different parameters for each layer. Sharing parameters between layers has been proposed before (Jastrzebski et al., 2017) (Boulch, 2017) and both papers report impressive model size reductions.

To reduce the computational cost we incorporate an Adaptive Computation Time (ACT) mechanism. In a traditional ResNet every sample follows the exact same path through the network where every layer refines the features from the previous layer. Not all samples are equally hard to classify and some of them will need more refinement steps than others. With ACT we can include a component that will decide how many steps are needed for each given input sample. ACT was introduced by Graves (2016) for recurrent neural networks where the network adapts the number of calculation

*The research work was done during an internship at NVIDIA research, Santa Clara, California, USA

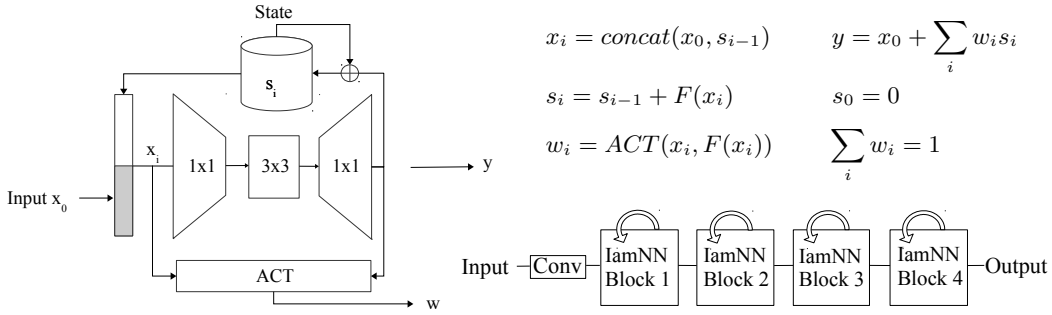

Figure 1: The IamNN network has the same structure as a ResNet but we replace the many residual units in each block by the architecture on the left which reuses the same weights multiple times.

steps to the complexity of the input. ACT can also be applied to networks for image recognition. Figurnov et al. (2016) introduced a ResNet based architecture where ACT was used to decide to evaluate or to skip certain layers. They even further improved this to Spatial ACT (SACT) which adapts the amount of computation between spatial positions.

We combined these two ideas of parameter sharing and adaptive computation time to design a building block for accurate deep neural networks with small model size and adaptive computational cost.

2 ARCHITECTURE

A typical ResNet consists of first a convolutional layer with batch normalization and ReLu nonlinearity followed by maxpooling. Then, a sequence of four blocks is stacked where each block consists of multiple stacked residual units. Each residual unit consists of one or more convolutional layers and a shortcut connection. All residual units in a block operate on the same spatial size. The first convolutional layer in each block uses a convolution with stride 2 to reduce the spatial size.

We propose to replace each block by the architecture shown in figure 1 (left). Each residual unit in a block is replaced by an iteration of this module. We use maxpooling between each block to reduce the spatial size. Our block consist of three main parts: a processing block with three convolutional layers, a state buffer where the results are accumulated and the ACT block that decides how many iterations of this block are needed.

The state buffer is used to gradually build the output of the block. The output of each iteration is added to the state which allows for an iterative refinement of the features. The initial state (s_0) is initialized with zero values.

The processing block consists of three convolutional layers with a bottleneck structure (He et al., 2016). At each iteration we concatenate the original input of the block (x_0) with the current state (s_{i-1}) before passing the concatenated vector through the processing block. The output of the processing block is added to the state. Each convolutional layer is followed by batch normalization, as is common in ResNets. We found that it is not possible to use the same batchnorm statistics for multiple iterations, instead we use a different set of statistics in each iteration. This only incurs a small overhead since the number of batchnorm statistics is very small compared to the full network. The same approach was used in Jastrzebski et al. (2017) in their experiments with shared weights.

The ACT block is a small two layer fully connected network (2 times 64 hidden units) that decides whether to keep evaluating the block or to move on to the next block.We concatenate the current state, the original input and the output of the processing block and apply global average pooling to obtain a vector with a single value for each channel. The ACT network uses this vector to calculate a *halting score* between 0 and 1 (sigmoidal activation).We sum the halting scores of each iteration and as soon as the cumulative halting score reaches one we stop evaluating this block and move on to the next block. We add an additional loss term to the classification loss to encourage the network to use a small number of iterations.

3 RESULTS

We trained our models on three benchmark datasets for image recognition: CIFAR10, CIFAR100 and ImageNet. We report the number of parameters, the theoretical number of operations and the test accuracy in table 3. We designed our models after a ResNet architecture and constrained them to use at maximum the same number of iterations in each block as the the number of units in that block of the ResNet. ResNet152 for example has four blocks with 3, 8, 36 and 3 residual units respectively. The corresponding IamNN is then constrained to use at maximum 3, 8, 36 and 3 iterations for each block respectively. The computational cost of the IamNN networks varies depending on the complexity of the input image, we report the average required FLOPS over all images in the test set.

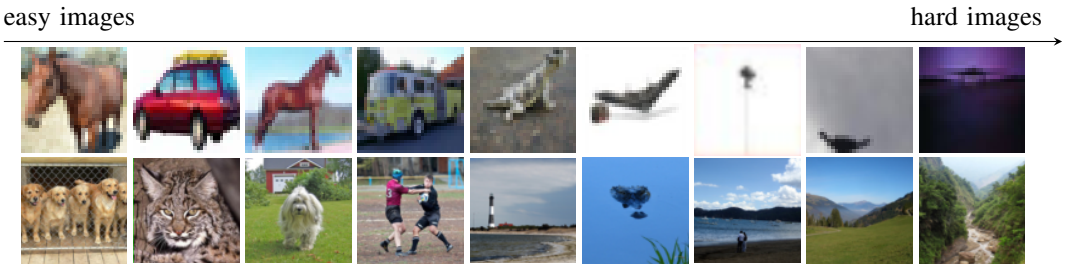

Figure 3: The spectrum of easy (left) to hard (right) to classify images for the network trained on CIFAR10 (top row) and ImageNet (bottom row). The network automatically adapts the computational cost to the complexity of the input image.

For CIFAR10 and CIFAR100 we reduce the number of parameters by 90% compared to the corresponding ResNet101. The number of FLOPS varies between 0.8 GFLOPs and 2 GFLOPs for both datasets depending on the complexity on the input image. On average we require 1.1 billion and 1.6 billion operations for CIFAR10 and CIFAR100 respectively, a reduction of 56% and 36% respectively. For CIFAR10 we obtain a slightly higher accuracy compared to the ResNet, probably because weight sharing acts as a regularizer for the small dataset. On CIFAR100 however the accuracy drops from 79.3% to 77.8%.

For the ImageNet dataset we used ResNet152 as a baseline. We are again able to reduce the number of parameters with 90% and the computations by 65% the top5 accuracy drops from 93.3% to 89%. To illustrate the benefit of iterative refinement using the same weights we also include the result when our IamNN network is only allowed one iteration per block. In this case there is no iterative refinement possible and the network obtains a top5 accuracy of 83.2%. We also compare to other architectures that were built to be very efficient in terms of computational cost and memory footprint (MobileNet (Howard et al., 2017), ShuffleNet (Zhang et al., 2017), SqueezeNet (Iandola et al., 2016) and GoogleNet (Szegedy et al., 2014)). These networks obtain a similar accuracy with a similar number of parameters but at a lower computational cost. This is because we did not change the basic operations of the network (1x1 and 3x3 convolutions). Some of the techniques used in the efficient networks such as depthwise separable convolutions are however orthogonal to our weight sharing and ACT approach and could be used to reduce the computational cost even further.

Figure 2 shows how many iterations are used for each block in the ImageNet network. The vertical line indicates the number of residual units in the corresponding ResNet (= the maximum number of iterations for the IamNN). Figure 3 illustrates how the network is able to adapt the computational cost to the complexity of the image. We show some typical images sorted from fast (left) to slow (right) to classify. The computation cost varies between 0.7G and 2G FLOPS for CIFAR10 and between 2.5G and 9G FLOPS for ImageNet.

Dataset	Network	Params	FLOPS	Top1/ Top5 (%)
CIFAR10	ResNet101	42 M	2.5G	93.8
	IamNN	4.5 M	1.1G (.7G - 2G)	94.6
CIFAR100	ResNet101	43 M	2.5G	79.3
	IamNN	4.6 M	1.6G (.7G - 2G)	77.8
ImageNet	ResNet152	60 M	11.5 G	77.0 / 93.3
	ResNet18	12 M	1.8 G	69.5 / 89.2
Single crop	IamNN 1 iter	4.8 M	0.9 G	60.8 / 83.2
Single network	**IamNN**	**5 M**	**4 B (2.5G - 9G)**	**69.5 / 89.0**
	ShaResNet34	14 M	11 G	71.0 / 91.5
	Googlenet	7 M	1.6 G	65.8 / 87.1
	MobileNet1	4.2 M	570 M	70.6 / 89.5
	ShuffleNet2x	5.6 M	524 M	70.9 / 89.8
	SqueezeNet	1.3 M	830 M	57.5 / 80.3

Table 1: Number of parameters and classification accuracy on three benchmark datasets for our IamNN architecture and the corresponding ResNet architecture.

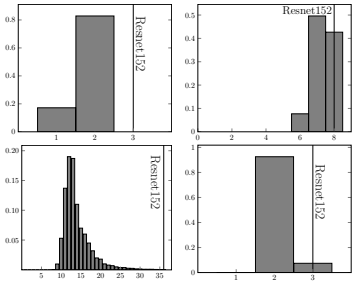

Figure 2: Number of iterations used in each of the four blocks (ImageNet network)

4 Conclusion and future work

We proposed a new ResNet based architecture based on insights in the iterative refinement behavior of ResNets and Highway networks. By using the same set of weights multiple times we can reduce the model size by 90%. Thanks to adaptive computation time the computation cost of the model depends on the complexity of the input image and is on average much lower than typical ResNets. In future work we will try to improve the results on the ImageNet dataset and we will incorporate other techniques such as depthwise separable convolutions to reduce the computational cost even further.

REFERENCES

Alexandre Boulch. Sharesnet: reducing residual network parameter number by sharing weights. *arXiv preprint arXiv:1702.08782*, 2017.

Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. *arXiv preprint arXiv:1612.02297*, 2016.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Klaus Greff, Rupesh K Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. *arXiv preprint arXiv:1612.07771*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Stanisław Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. *arXiv preprint arXiv:1710.04773*, 2017.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL `http://arxiv.org/abs/1409.4842`.

Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, pp. 550–558, 2016.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.